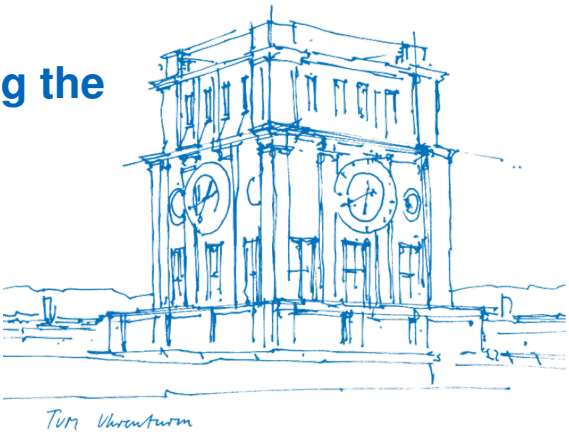
Chair of Computer Architecture and Parallel Systems
TUM School of Computation, Information and Technology
Technical University of Munich

Optimizing Neutral Atom Rearrangement Times using the HiPARS Sorting Library

Jonas Winklmann (TUM)
Martin Schulz (TUM)

Chair of Computer Architecture and Parallel Systems
TUM School of Computation, Information and Technology
Technical University of Munich

February 23rd, 2026



Acknowledgment

MUNIQC-Atoms
Neutral Atom based
Quantum Computing Demonstrator

Munich Quantum Valley

Federal Ministry
of Research, Technology
and Space

This work was funded by the German Federal Ministry of Research, Technology and Space (BMFTR) under the funding program *Quantum Technologies - From Basic Research to Market* under contract number 13N16087, as well as from the Munich Quantum Valley (MQV), which is supported by the Bavarian State Government with funds from the Hightech Agenda Bayern.

Outline

Neutral Atom Quantum Computing

Setup at MPQ

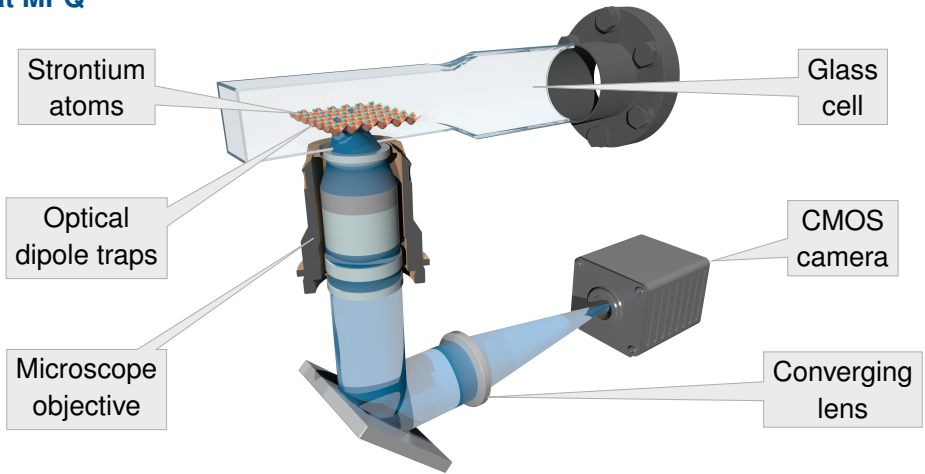
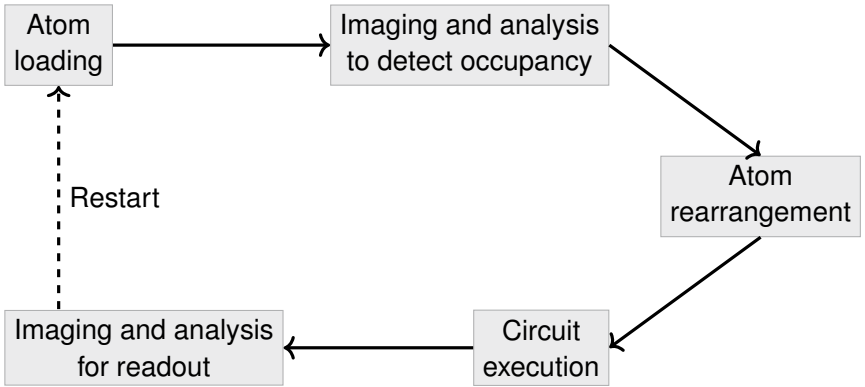


Illustration by Dr. Andrea Alberti [Winklmann et al., 2023]

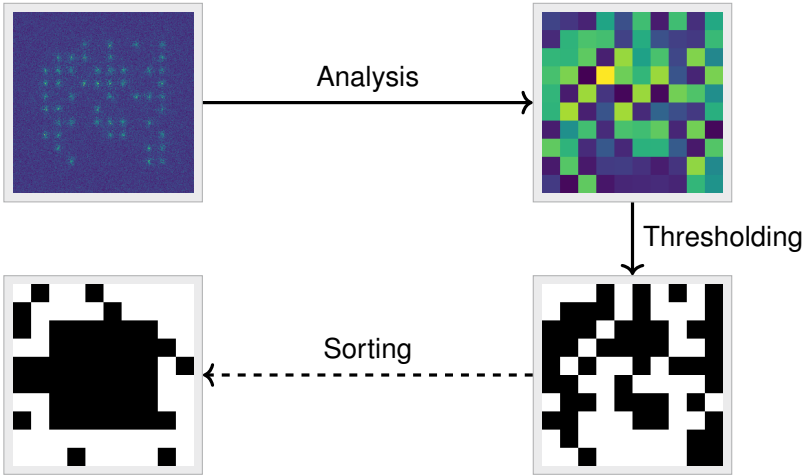
Neutral Atom Quantum Computing

Compute Cycle



Neutral Atom Quantum Computing

Imaging and Analysis

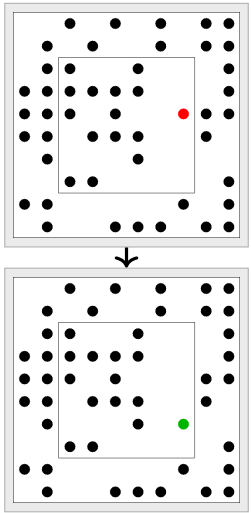


Neutral Atom Quantum Computing
Sorting

Sorting time may represent significant fraction of cycle time

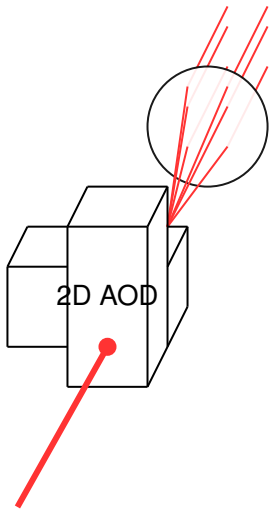
- Time per move $\approx 0.1ms - 1ms$
- Number of moves $\propto N^{1.16}$ for N target sites [Tian et al., 2022]

➤ Sorting times up to several 100ms for few hundred qubits



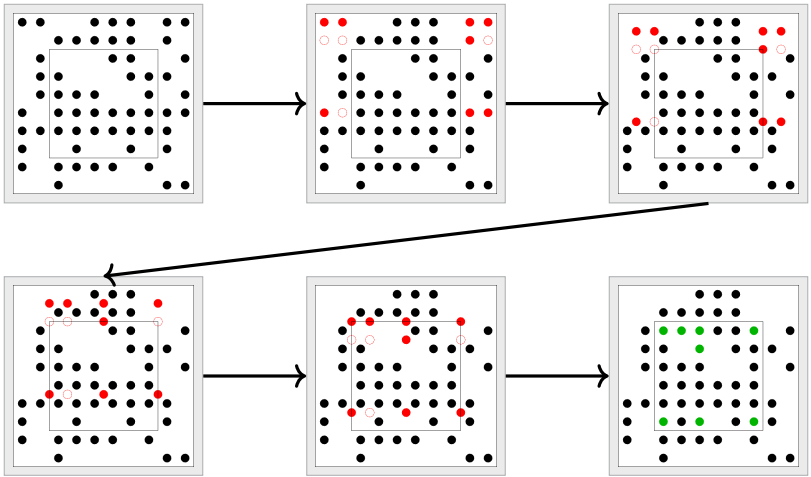
Neutral Atom Quantum Computing
Parallelized Sorting

- Address atoms in parallel
- Two AODs that can be fed frequencies independently
- Selection of rows $f_h = \{f_{h,1}, ..., f_{h,i}\}$ and columns $f_v = \{f_{v,1}, ..., f_{v,j}\}$ produces traps at Cartesian product $f_h \times f_v$.



Neutral Atom Quantum Computing

Parallelized Sorting



Outline

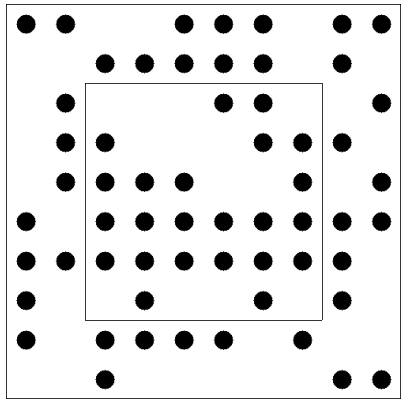
Parallel Tweezer Sorting Algorithm

- Input
  - Initial state array
  - Target geometry
  - Various setup parameters
- Returns list of moves that produce desired target state
- Greedy approach that executes best move at each step
- C++ library with Python wrapper
- Excels with unconstrained movement between rows and columns

Parallel Tweezer Sorting Algorithm

Move

Sequence of frequencies for the horizontal and vertical AOD.



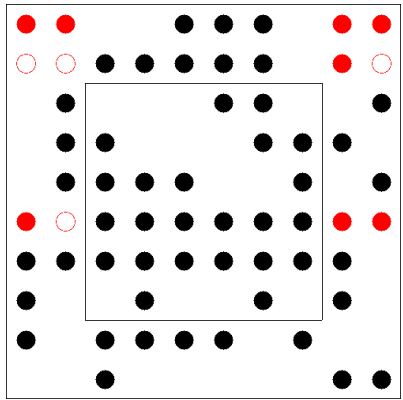
Parallel Tweezer Sorting Algorithm

Move

Sequence of frequencies for the horizontal and vertical AOD.

Ramp up starting frequencies:

- Rows: {0, 1, 5}
- Columns: {0, 1, 8, 9}
- Ramp up amplitude



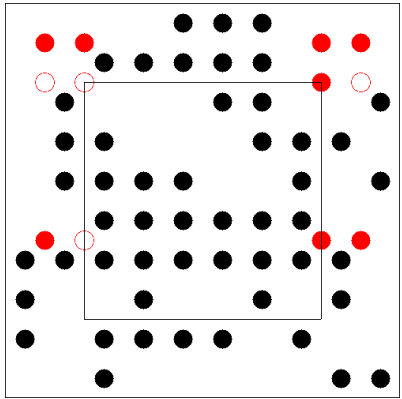
Parallel Tweezer Sorting Algorithm

Move

Sequence of frequencies for the horizontal and vertical AOD.

First step:

- Rows: {0.5, 1.5, 5.5}
- Columns: {0.5, 1.5, 7.5, 8.5}



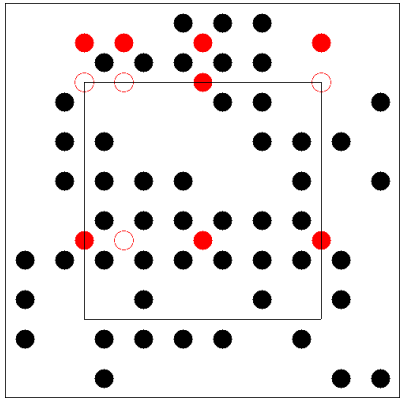
Parallel Tweezer Sorting Algorithm

Move

Sequence of frequencies for the horizontal and vertical AOD.

Second step:

- Rows: {0.5, 1.5, 5.5}
- Columns: {1.5, 2.5, 4.5, 7.5}



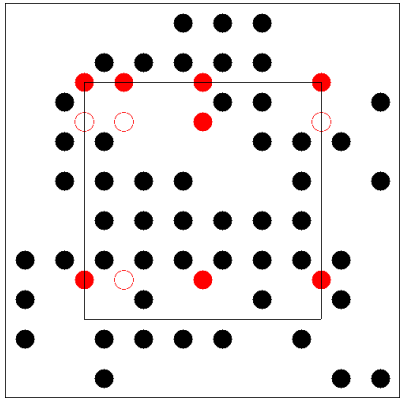
Parallel Tweezer Sorting Algorithm

Move

Sequence of frequencies for the horizontal and vertical AOD.

Third step:

- Rows: {1.5, 2.5, 6.5}
- Columns: {1.5, 2.5, 4.5, 7.5}



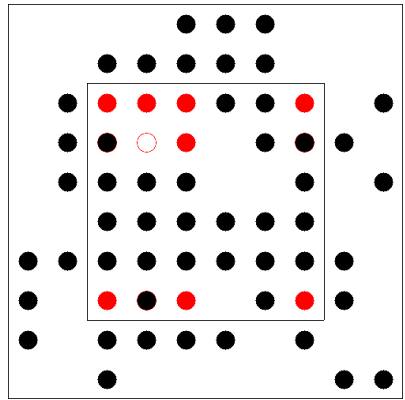
Parallel Tweezer Sorting Algorithm

Move

Sequence of frequencies for the horizontal and vertical AOD.

Four step:

- Rows: {2, 3, 7}
- Columns: {2, 3, 4, 7}



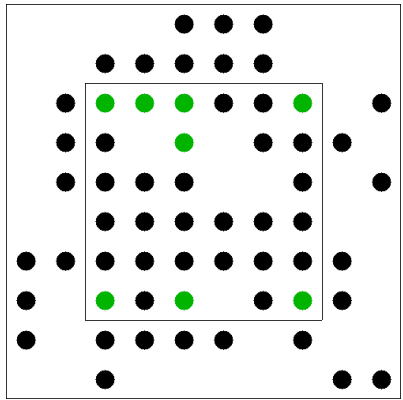
Parallel Tweezer Sorting Algorithm

Move

Sequence of frequencies for the horizontal and vertical AOD.

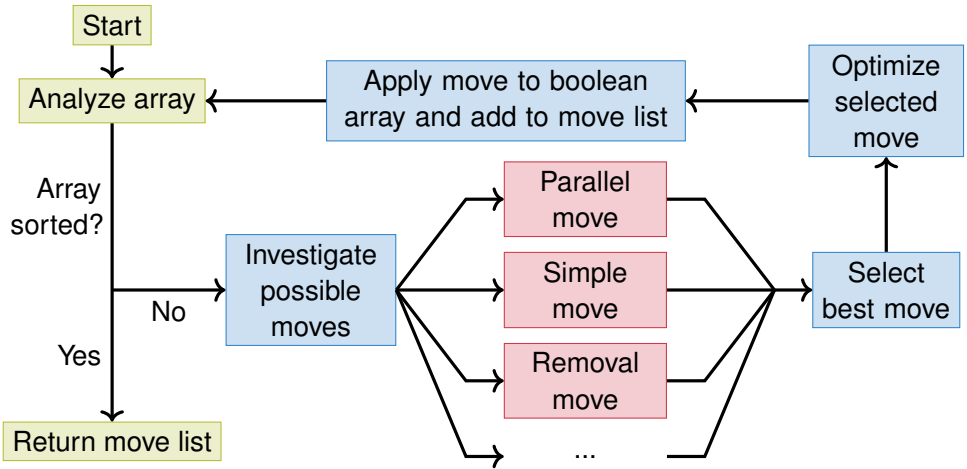
After last step:

- Rows: {2, 3, 7}
- Columns: {2, 3, 4, 7}
- Ramp down amplitude



Parallel Tweezer Sorting Algorithm

Workflow



Parallel Tweezer Sorting Algorithm
Fitness function

How to select the "best" move?

- Most target sites filled per time spent
- Requires configurable function $t_m(d)$ that describes time to execute move

We offer four configuration options c, c_s, l, r:

$$t_m(d) = c + \sum_{i=0}^{n} (c_s + d_i \cdot l + \sqrt{d_i} \cdot r) \tag{1}$$

where d_i describes the maximum distance that any traps moves during step i.

Parallel Tweezer Sorting Algorithm
Move Optimization

- Adding new row or column that improves move
  - Relatively simple and fast, can improve any type of move

- Adding an independent atom by adding a row and column
  - Slower, thus limited to only independent atoms
  - Improves sorting towards the end

Video not supported. Consider using Acrobat Reader or Okular

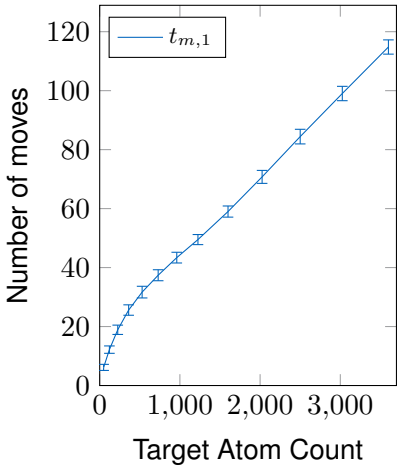
Parallel Tweezer Sorting Algorithm

Results - Move Count

Average move counts with default configuration:

- time-demand functions $t_{m,1}(d) = 120\mu s + \frac{d}{0.13\frac{\mu m}{\mu s}}$ and $t_{m,2}(d) = 120\mu s + \frac{d}{0.55\frac{\mu m}{\mu s}}$
- an initial filling ratio $f = 0.5$
- the ratio between total and target size $r = 1.5$
- a limit to the number of frequencies for both vertical and horizontal AOD of $n_h = n_v = 16$
- a limit to the total number of generated movable traps $k = 256$

Every datapoint is generated using 1000 simulations.
Error bars denote $\pm$ one standard deviation.

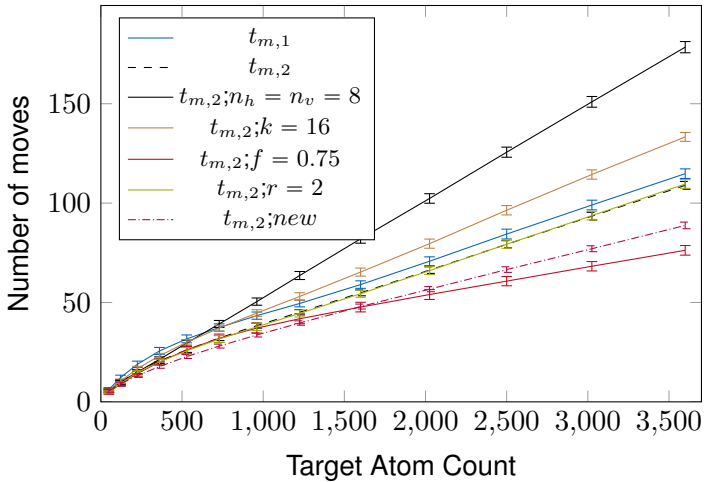


Parallel Tweezer Sorting Algorithm

Results - Move Count

Average move counts that our rearrangement algorithm generates given different inputs

Error bars denote $\pm$ one standard deviation.

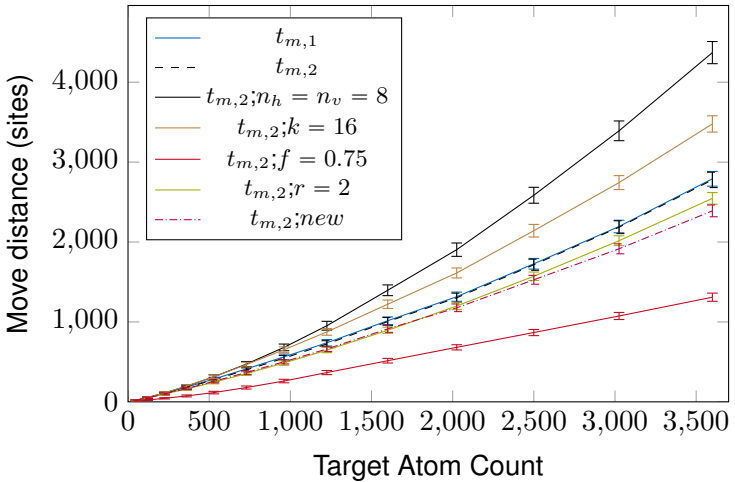


Parallel Tweezer Sorting Algorithm

Results - Total Move Distance

Average total move distance that our rearrangement algorithm generates given different inputs

Error bars denote $\pm$ one standard deviation.

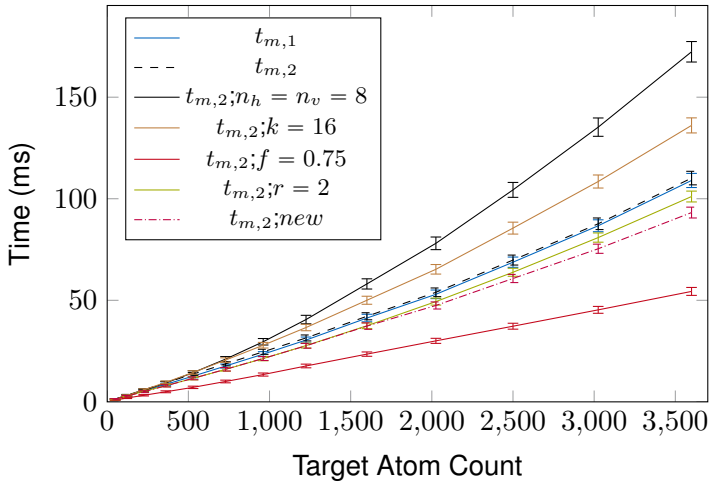


Parallel Tweezer Sorting Algorithm

Results - Move Execution Time

Average required move execution time using $t_{m,1}(d)$ to simulate execution time.
Results may vastly differ for different setups.

Error bars denote $\pm$ one standard deviation.

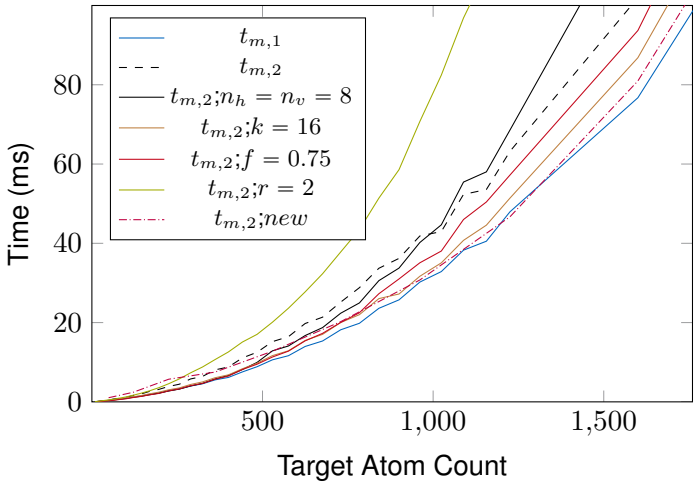


Parallel Tweezer Sorting Algorithm

Results - Run Time

Average algorithm run time

Algorithm run time currently somewhat limiting



Parallel Tweezer Sorting Algorithm

Results - Success rate

- Algorithm is guaranteed to succeed if enough atoms are present

Parallel Tweezer Sorting Algorithm

Results - Success rate

- Algorithm is guaranteed to succeed if enough atoms are present
- Main source of atom loss typically trap transfer
- ➤ Algorithm already produces few moves

Parallel Tweezer Sorting Algorithm

Results - Success rate

- Algorithm is guaranteed to succeed if enough atoms are present
- Main source of atom loss typically trap transfer
- ➤ Algorithm already produces few moves
- ➤ Should fitness function include fidelity?

Outline

Comparison
AOD

Compared to best found AOD-based alternatives

Under favorable conditions at 400 atoms

- ≈33% fewer moves
- ≈50% faster move execution
- No other approach shows advantage at any atom count but our run time limits number of qubits

Comparison
SLM

Compared to best found SLM-based alternatives

- Constant 60ms for 1000-10000 atoms [Lin et al., 2024]
- Our approach is faster up to around 1000 atoms
- Benefit in move execution time up to around 2000 atoms
- Further SLM hardware improvements to be expected

Outline

HiPARS

Highly-Parallel Atom Rearrangement Sequencer

How to efficiently rearrange atoms?

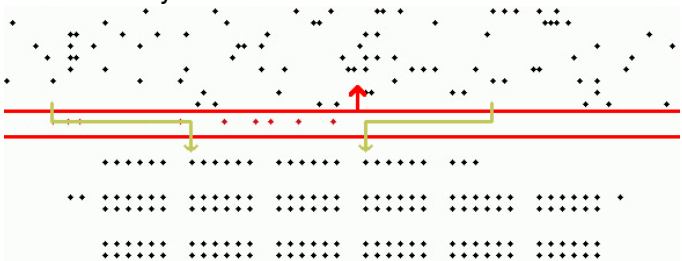
- Complicated holistic approach can be too slow for simple scenarios
- ➤ Comprehensive sorting library with specialized solutions for different cases

Currently supported:

- ✓ Simple sequential sorting towards fully-filled target
- ✓ Greedy parallel sorting towards arbitrary target geometries
- ✓ Fast parallel sorting for lattice-based setups towards arbitrary target geometries
- × Different geometries (triangle, honeycomb, ...)

HiPARS
Lattices

- Much higher atom count
- Much smaller spacings
- ➤ Different constraints
- ➤ Some atoms may not even be separable reliably
- ➤ Greedy approach too complicated
- ➤ Stricter by-row movement scheme



Outline

Future Work

Concerning the parallel tweezer algorithm:

- More sophisticated fitness function
- Different move types and/or system to only investigate certain moves based on configuration/rearrangement stage

Generally:

- Different target geometries
- Support even more constraints

Conclusion

- Outperforms current AOD-based sorting approaches significantly for certain tweezer setups
- Advantage over SLMs on current technology and qubit numbers
- Run time can be a drawback
- More constraints, arbitrary target configurations and geometries to be supported in the future

✉ jonas.winklmann@tum.de

ce.cit.tum.de/caps

HiPARS

MUNIQC-Atoms
Neutral Atom based
Quantum Computing Demonstrator

Munich Quantum Valley